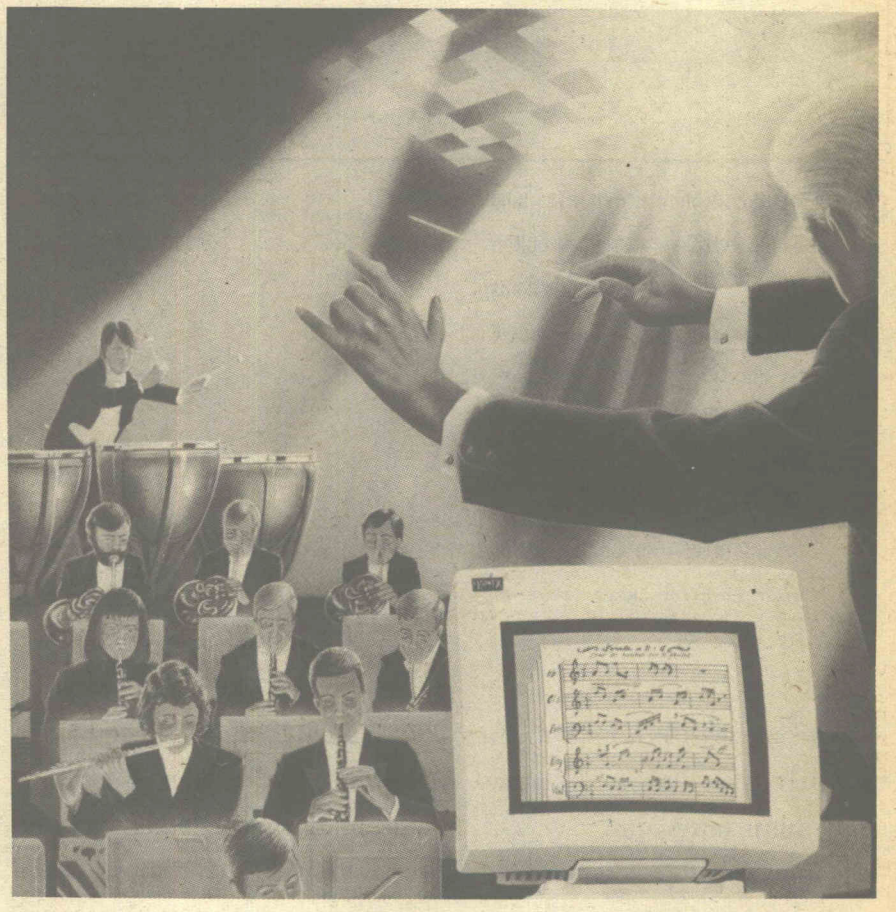
SOFT MAIL

เทอร์โบปาสคาลกับวินโดวส์ : คอนโทรล

เพิ่มรูปแบบและประสิทธิภาพในการใช้งานวินโดวส์กับการเขียนโปรแกรมอย่างมีระบบ



สิ่งที่เห็นในไดอะล็อกบ็อกซ์ คือ คอนโทรล (Control) จนอาจกล่าวได้ว่าในไดอะล็อกบ็อกซ์มีแต่คอนโทรล และอาจกล่าวได้อีกว่าการเขียนโปรแกรมบนวินโดวส์คือ การเขียนโปรแกรมใช้งานคอนโทรล โดยที่คอนโทรลไม่จำเป็นต้องอยู่ในไดอะล็อกบ็อกซ์เสมอไปอาจอยู่ในหน้าต่างได้เช่นกัน

นานาชนิดของคอนโทรล

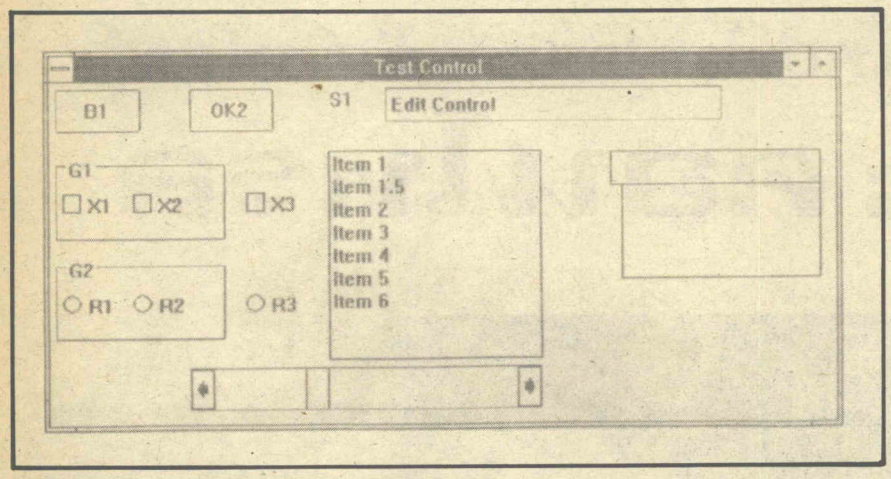
คอนโทรลมีอยู่มากมายหลายชนิด แต่ที่เป็นหลัก ๆ มีอยู่ด้วยกัน 9 ชนิด ดังแสดงในรูปที่ 1 (และสำหรับในเทอร์โบปาสคาลได้รวมเอา IMDIClient ตามที่กล่าวในไมโครฉบับที่ 84 มาเป็นคอนโทรลด้วยอีกชนิดหนึ่ง) ที่ผมกล่าวว่าเป็นแบบหลัก ๆ เพราะคอนโทรลบางชนิดสามารถแตกเป็นชนิดย่อยได้อีก หรือนำมารวมกันเป็นชนิดใหม่

การเรียกชื่อคอนโทรลทั้ง 9 ชนิดนี้เรียกไม่ค่อยเหมือนกันคือ ในคู่มือการใช้งานวินโดวส์เรียกอย่างหนึ่ง ในคู่มือการเขียนโปรแกรมบนวินโดวส์ซึ่งก็เป็นของไมโครซอฟต์เรียกไปอีกอย่างหนึ่ง และสำหรับการเขียนโปรแกรมด้วยเทอร์โบปาสคาลยังมีชื่อสกุลของคอนโทรลเข้ามาเกี่ยวข้องอีกด้วย ผมจึงให้ชื่อที่กำหนดเรียกในคู่มือการเขียนโปรแกรมของไมโครซอฟต์ ไว้ในรูปที่ 1 ซึ่งในเทอร์โบปาสคาลใช้ตามนี้ด้วย สำหรับชื่อในคู่มือการใช้ของดกล่าวถึงส่วนหน้าที่การใช้งานคอนโทรลแต่ละชนิดซึ่งไมโครซอฟต์ได้กำหนดไว้เด่นชัดและเราจะต้องใช้ตามนี้ด้วย จะขอนำไปกล่าวรวมกับคำอธิบายสกุล

สกุลของคอนโทรล

ตามที่เคยกล่าวมาแล้วว่าเทอร์โบปาสคาลได้นำหน้าต่างทุกชนิดมาทำเป็นออปเจ็กต์ โดยสร้างเป็นสกุลไว้ สำหรับพวกคอนโทรลได้สร้างสกุล TControl เป็นสกุลต้นตระกูลของคอนโทรลทั้งหลายและเป็นสกุลโครง (abstract) คือไม่ใช่สำหรับการกำหนดอินสแตนซ์นำไปใช้งาน

จากสกุล TControl ได้สร้างผู้สืบสกุลเป็นสกุลของคอนโทรลต่าง ๆ และยังสืบสกุลกันเองอีกด้วย โดยที่ผมจะให้ชื่อภาษาไทยไว้ในฐานะออปเจ็กต์



รูปที่ 1 แสดงลักษณะของคอนโทรลชนิดต่าง ๆ ได้แก่

- B1 และ OK2 คือ Push bulton
- S1 คือ Staltic control
- Edit คือ Edit control
- G1 และ G2 คือ Group box
- X1, X2 และ X3 คือ Check box
- R1 และ R2 คือ Radio bulton
- กรอบที่มีคำว่า Item คือ List box
- กรอบเปล่าทางขวา คือ Combo box (แบบหนึ่ง)
- ล่างสุด คือ Scroll bar

ตามด้วยชื่อสกุลเป็นภาษาอังกฤษ สำหรับชื่อภาษาไทยจะใช้คำว่าปุ่มกับ buttons และกรอบกับ boxes คือ

ปุ่มควบคุม สกุล TButton (ตามรูปที่ 1 คือ push button) เป็นปุ่มที่ใช้สำหรับควบคุมการทำงานของโปรแกรม เช่น ปุ่ม OK ซึ่งเมื่อผู้ใช้เลือก (คลิ๊กหรือใช้คีย์) จะมีการปฏิบัติทันที (คือต้องเขียนโปรแกรมให้เป็นไปตามนี้)

ปุ่มเช็ค สกุล TCeckBox เป็นปุ่มสี่เหลี่ยมที่ให้ผู้ใช้เลือก คือเซตไว้สำหรับการดำเนินการหนึ่ง ๆ ส่วนมากจะจัดไว้เป็นกลุ่มอยู่ในกรอบกลุ่ม ผู้ใช้สามารถเลือกได้หลายปุ่มในกรอบเดียวกัน เช่น ผมจะทำเป็นรายการผักสำหรับสลัดให้ผู้ใช้เลือกว่าลูกค้าเลือกผักใดบ้างสำหรับสลัดจานหนึ่ง ๆ

ปุ่มวิทยุ สกุล TRadioBulton สืบสกุลจาก TCheckBox อีกทอดหนึ่ง เป็นปุ่มกลมที่ให้ผู้ใช้เลือกคือเซตเช่นกัน และโดยปกติจะต้องจัดเป็นกลุ่ม ให้ผู้ใช้เลือกปุ่มใดปุ่มหนึ่งในหลายปุ่ม และเมื่อเซตปุ่มใดปุ่มอื่นจะถูกรีเซตโดยอัตโนมัติ เช่นตามรูปที่ 1 ถ้าเซตปุ่ม R1 ปุ่ม R2 จะถูกรีเซต

กรอบกลุ่ม สกุล TGroupBox เพื่อรวมปุ่มเลือกหรือปุ่มตรวจสอบแต่ละชนิดไว้ด้วยกัน ตามรูปที่ 1 คือกรอบ G1 และ G2 จะทำหน้าที่เป็นตัวแทนของปุ่มต่าง ๆ ในกรอบในการติดต่อกับเราคือ หน้าต่าง

กรอบหัวข้อ สกุล TStatic ส่วนมากจะใช้เป็นหัวข้อของกรอบป้อน ขอให้พลิกไปดูในรูปที่ 4 จะเห็นถนัดกว่าเช่น กรอบ Name, Address1 และที่ตามมาเป็นกรอบที่ไม่แสดงตัวกรอบ

กรอบอีดิต สกุล TEdit สืบสกุลจาก TStatic อีกทอดหนึ่ง เพื่อให้ผู้ใช้ป้อนข้อความกรอบละบรรทัดเดียว เช่นตามรูปที่ 4 มีหลายกรอบป้อนอยู่ทางขวาของกรอบหัวข้อ หรือให้ป้อนหลายบรรทัดก็ได้ คือจะให้ทั้งหน้าต่างมีแค่กรอบอีดิตเพียงกรอบเดียวได้เช่นกัน

กรอบรายการ สกุล TListBox เป็นกรอบสำหรับแสดงรายการให้ผู้ใช้เลือกที่ใช้กันมาคือรายการชื่อไฟล์ โดยเฉพาะชื่อไดเรคทอรี เมื่อเราสร้างกรอบชนิดนี้แล้วจะต้องให้รายการที่จะให้ผู้ใช้เลือก ซึ่งกรอบนี้จะนำไปแสดงในแบบเรียงลำดับอักษร แต่สามารถกำหนดให้เรียงตามลำดับก่อนหลังได้ และสามารถกำหนดแบบย่อยเป็นแบบที่จะให้ผู้ใช้เลือกเพียงรายการเดียวหรือหลายรายการ

กรอบคอมโบ สกุล TComboBox เป็นกรอบที่ผสมกรอบป้อนกับกรอบรายการเพื่อให้ผู้ใช้สามารถทั้งป้อนหรือเลือกรายการได้ เช่นการเลือกรายการไฟล์ แบ่งออกเป็น 3 ชนิดคือ

- Simple combo คือกรอบคอมโบชนิดธรรมดาจะแสดงส่วนที่เป็นกรอบรายการตลอดเวลา

- Drop down combo เช่นเดียวกับกรอบ Simple combo แต่จะแสดงกรอบรายการเมื่อผู้ใช้เลือกคือเลือกปุ่มที่อยู่ทางขวาของกรอบป้อน

- Drop down list combo เช่นเดียวกับกรอบ Drop down combo คือจะแสดงกรอบรายการเมื่อผู้ใช้ต้องการแตกต่างกันตรงการป้อนข้อความในกรอบป้อนคือ เมื่อผู้ใช้ป้อนข้อความตรงกับรายการหนึ่งรายการใดแล้วจะป้อนต่อไม่ได้

ทำความเข้าใจกับกรอบทั้งสามชนิดนี้กับกรอบรายการได้ด้วยโปรแกรมในไฟล์ CBOXTEST.PAS (โปรแกรมในเทอร์โบปาสคาลตามที่ผมจะอ้างถึงในบทความตอนนี้ทั้งหมดอยู่ในไดเรคทอรี C: \ TPW \ DOCDEMOS)

สโครลบาร์ สกุล TScrollBar เป็นสโครลบาร์อิสระ และเป็นคนละตัวกับสโครลบาร์ที่กำกับกรอบของหน้าต่าง (รวมทั้งกรอบรายการทุกชนิดด้วย) ซึ่งเป็นของสกุล TScroller และไม่ได้จัดอยู่ในพวกคอนโทรล สำหรับสโครลบาร์ (อิสระ) เพื่อให้ผู้ใช้ป้อนข้อมูลต่อเนื่อง เช่นการผสมสี ซึ่งผมจะขอเว้นไม่นำมาทำตัวอย่าง ขอให้ดูตัวอย่างการสร้างและการนำมาใช้ได้ในไฟล์ SBARTEST.PAS ของเทอร์โบปาสคาล

คอนโทรลสองประเภท

คงจะพอสำหรับการแนะนำให้รู้จักกับคอนโทรลครับ ซึ่งคอนโทรลทั้งหมดนี้เทอร์โบปาสคาลได้แบ่งออกเป็น 2 ประเภทคือ ที่ใช้ในหน้าต่างได้กำหนดให้ใช้ในฐานะออปเจ็กต์ ส่วนที่อยู่ในไดอะล็อกบ็อกซ์ไม่เป็นออปเจ็กต์ ดังนั้นในคู่มือสกุลของเทอร์โบปาสคาลจึงกล่าวว่าปกติใช้ในหน้าต่างไม่ใช้ในไดอะล็อกบ็อกซ์ หมายถึงคอนโทรลที่เป็นออปเจ็กต์

สำหรับในบทความตอนนี้จะกล่าวถึงการใช้คอนโทรลในหน้าต่างคือ การใช้คอนโทรลในฐานะออปเจ็กต์ ต้องขอความกรุณาให้งดคิดถึงไดอะล็อกบ็อกซ์ชั่วคราวก่อน

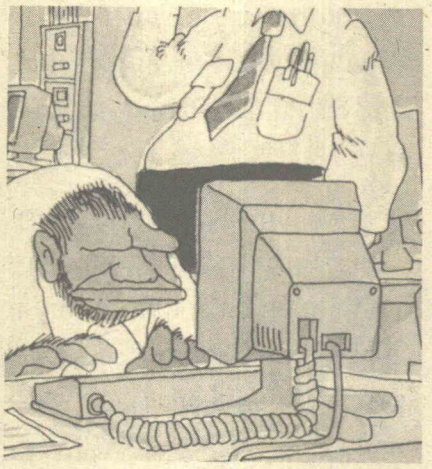
สื่อความกับเมสเสจ

ผมใช้คำว่า "สื่อความ" กับคำในภาษาอังกฤษว่า "message" ตลอดมา แต่ตอนนี้ต้องขอแยกครับ เพราะจะเริ่มเขียนโปรแกรมกันจริงจัง โดยที่การเขียนโปรแกรมบนวินโดวส์ด้วยเทอร์โบปาสคาล คำว่า message มีความหมายอยู่ 2 นัยคือ

- เป็น message ของวินโดวส์ ซึ่งถึงแม้จะอยู่ในรูปของตัวอักษร เช่น wm__Paint ก็เป็นค่าคงตัวซึ่งมีค่าเป็นเลขจำนวนเต็มบวก (Word)

- เป็น message ทางด้านโอโอพีของเทอร์โบปาสคาล เช่น appl.Run

ในหนังสือภาษาอังกฤษก็ต้องคอยแยกครับ เพื่อที่ผมจะได้ไม่ต้องเขียนข้อความกำกับอยู่ตลอดเวลา จึงขอใช้คำว่าเมสเสจกับ message ของวินโดวส์ และสื่อความกับ message ของเทอร์โบปาสคาล



ถึงเป็นเมสเสจก็ไม่เหมือนกัน

ถึงแม้ผมจะแยกว่าเมสเสจเป็นของวินโดวส์ แต่เมื่อเขียนโปรแกรมด้วยเทอร์โบปาสคาลกลับแตกต่างกันอีก

วินโดวส์ ขอกล่าวถึงทางซีกของวินโดวส์ก่อนครับ เราสามารถแยกเมสเสจของวินโดวส์ออกได้เป็น 3 กลุ่มคือ

1. กลุ่มแรก เป็นเมสเสจจากวินโดวส์ถึงหน้าต่าง รวมทั้งคอนโทรลเพราะเป็นหน้าต่างเหมือนกัน เป็นเมสเสจที่ขึ้นต้นด้วยอักษร wm__ (คงหมายถึงเมสเสจถึงหน้าต่าง) เช่น wm__LButtonDown

2. กลุ่มที่สอง เป็นเมสเสจจากหน้าต่างถึงไดอะล็อกบ็อกซ์และคอนโทรล โดยจัดให้อักษรนำมีความหมายถึงผู้รับ คือ bm__, cb__, dm__, em__ และ lb__ หมายถึง botton ทุกชนิด, combo box, dialog bog, edit control และ list box ตามลำดับ

3. กลุ่มที่สาม เป็นเมสเสจที่เกิดขึ้นเมื่อผู้ใช้เลือกรายการในเมนู หรือเลือกคอนโทรล หรือเมื่อแปลค่าคีย์ด่วน จึงมองเป็นเมสเสจที่มาจากสิ่งต่าง ๆ เหล่านี้ มีอยู่เมสเสจเดียวคือ wm__Command โดยที่เราคือผู้เขียนโปรแกรมให้แก่หน้าต่างจะต้องตีความในเมสเสจจากฟิลด์ต่อไปนี้ในเมสเสจคือ

- wParam เป็นข้อมูล 2 ไบต์ เก็บค่ารายการของเมนู (ตามที่กล่าวในไมโคร ฯ 82) หรือค่าไอดีของคอนโทรล (ตามที่จะกล่าวในฉบับนี้) หรือค่าไอดีของคีย์ด่วน (ยังไม่ทราบครับว่าจะนำมากล่าวในฉบับใด)

- lParam เก็บข้อความที่ฝากส่งมา ซึ่งถ้าเขียนเป็นคำอธิบายจะยาวและเข้าใจยาก จึงเขียนเป็นรูปของโปรแกรมตามรูปที่ 2 โดยที่ lParam เป็นข้อมูลขนาด 4 ไบต์ Lo หมายถึง lParmLo คือสองไบต์ต่ำ Hi หมายถึง lParmlti คือสองไบต์สูงสำหรับความหมายของ notification code คือเหตุที่เกิดขึ้น ทำไว้เป็นค่าคงตัวเช่นกันขึ้นต้นด้วยอักษร bn__, en__, lbn__ และ cbn__ ซึ่งหมายถึง Button, Edit, List box และ Combo box ตามลำดับ ส่วน n หมายถึง notification codes

ค่าคงตัวของเมสเสจและโนติฟิเคชันโค้ดดังกล่าวมานี้ ดูได้จากคู่มือหรือ Help โดยค้นหาด้วยอักษรที่ขึ้นต้นดังกล่าว

และต้องขอทำความเข้าใจเพิ่มอีกเล็กน้อยคือ ถ้าเราเขียนโปรแกรมด้วยภาษาซีเมสเสจที่ได้รับคือ เมสเสจที่เป็นเรคอร์ดมีแบบข้อมูลเป็น TMsg ซึ่งในเรคอร์ดนี้มีหลายฟิลด์ที่ใช้ในการเขียนโปรแกรมเป็นประจำคือ ฟิลด์ wParam กับ lParm

อีกฟิลด์หนึ่งคือ message เป็นเมสเสจที่มีค่าเป็นเลขจำนวนเต็มบวกและได้ทำเป็นค่าคงตัวไว้ดังกล่าวมา ซึ่งในโปรแกรมภาษาซีจะต้องตรวจว่าเป็นค่าอะไร และดูรายละเอียดได้ใน wParam และ lParam ซึ่งจะแตกต่างกันไปในแต่ละเมสเสจ ค่าที่กล่าวในข้อ 3 คือของ wm__Command

เทอร์โบปาสคาล

เมื่อมาถึงเทอร์โบปาสคาล สิ่งแรกคือได้เปลี่ยนแบบข้อมูล TMsg เป็น TMessage เพื่อให้อ้างถึงข้อมูลในแต่ละฟิลด์ได้สะดวกยิ่งขึ้น เช่นใน TMsg มีฟิลด์ lParm ใน TMessage แยกเป็น lParamHi และ lParamLo

เรื่องที่สองคือได้แกะกล่องเมสเสจที่เป็นเรคอร์ดนำค่า message ออกมาทำเป็นตาราง MDT (ไมโครฯ 80 หน้า 270) การเขียนโปรแกรมด้วยเทอร์โบปาสคาลจึงแตกต่างกันเมื่อเขียนด้วยซี คือของเทอร์โบปาสคาลเทียบได้กับการดักจับเมสเสจที่เป็นเลขจำนวน

ดังเช่นกิจกรรมแรกในรูปที่ 3 เป็นการกำหนดกิจกรรมสำหรับดักจับเมสเสจ wm__LBultonDown ซึ่งจะเกิดขึ้นเมื่อผู้ใช้กดปุ่มซ้ายของเม้าส์ ขอให้สังเกตฐานคือ wm__First ไว้ด้วย และให้เราดำเนินการกับเมสเสจของวินโดวส์ทั้ง 3 กลุ่มดังนี้

1. กลุ่มแรก เป็นไปตามที่กล่าวข้างบนนี้คือดักจับเมสเสจ

2. กลุ่มที่สอง เทอร์โบปาสคาลให้เราใช้สื่อความ (ไม่ใช่เมสเสจ) ซึ่งเตรียมไว้ให้แล้วกับคอนโทรลในฐานะออปเจ็กต์

3. กลุ่มที่สาม เทอร์โบปาสคาลแกะกล่องอีกครับ โดยนำค่าใน wParam มาทำเป็นเมสเสจเลขจำนวนด้วยวิธีเดียวกับที่ทำกับ message แท้ ๆ คือด้วยตาราง MDT เรื่องนี้เป็นเรื่องที่แตกจากของวินโดวส์โดยสิ้นเชิง เพราะค่าใน wParam ไม่ใช่เมสเสจ ตามนัยของวินโดวส์เป็นค่าที่เราผู้เขียนโปรแกรมกำหนดขึ้นแต่ต้องใช้อย่างเมสเสจ โดยที่เทอร์โบปาสคาลได้กำหนดฐานไว้แตกต่างกันคือ

- จากเมนู ให้ฐานเป็น cm__First ค่านี้เทอร์โบปาสคาลเป็นผู้กำหนด กิจกรรมที่สองในรูปที่ 3 เป็นการกำหนดกิจกรรมไว้ดักจับเมสเสจ cm__One ซึ่งจะเกิดขึ้นเมื่อผู้ใช้เลือกรายการ One จากเมนู

- จากคอนโทรล ให้ฐานเป็น id__First กำหนดไว้ในเทอร์โบปาสคาลอีกเช่นกัน เช่น กิจกรรมที่สามในรูปที่ 3 เพื่อดักจับเมสเสจ id__Two ซึ่งเกิดขึ้นเมื่อผู้ใช้เลือกปุ่ม Two

ค่าของ cm__One กับ cm__Two เราเป็นผู้กำหนด เช่นอาจให้เป็น 200 ทั้งคู่ได้ เพราะฐานต่างกัน แต่ถ้าเป็นพวกเดียวกันจะต้องไม่ซ้ำกัน และให้ค่าได้ตั้งแต่ 100 ถึง 4096

ใช้งานคอนโทรล

การใช้งานคอนโทรลจะมีการดำเนินการอยู่ 5 เรื่องคือ

- ให้กำเนิดคอนโทรล
- ให้ค่าเริ่มต้นแก่คอนโทรล
- ดักจับเมสเสจจากคอนโทรล
- ควบคุมคอนโทรล
- อ่านและให้ค่าแก่คอนโทรล

ซึ่งในแต่ละขั้นตอนยังมีรายละเอียดซึ่งผมจะขอนำไปกล่าวประกอบกับตัวอย่าง

โปรแกรมตัวอย่าง

เพื่อให้เห็นการเขียนโปรแกรมบนวินโดวส์โดยส่วนรวมด้วยจะขอยก

รูปที่ 2
ค่าใน lParamHi คือ HI และ lParamLo คือ LO

```
if      LO = 0  then  Message is from a menu
else if HI = 1  then  LO is an accelerator ID
else
  begin LO is a handle to control;
        HI is a notification code  end;
```

รูปที่ 3
กิจกรรมสำหรับเมสเสจ 3 ชนิด

```
procedure  WMLButtonDown(var Msg : TMessage); virtual
                         wm_First + wm_LButtonDown;

procedure  CMOne(var Msg : TMessage); virtual
                   cm_First + cm_One;

procedure  IDTwo(var Msg : TMessage); virtual
                   id_First + id_Two;
```

ตัวอย่างเป็น ต้องการเขียนโปรแกรมให้ผู้ใช้ป้อนข้อมูลในนามบัตร เก็บลงไฟล์ และให้ผู้ใช้เรียกดูและแก้ข้อมูลได้ เป็นโปรแกรมที่ดำเนินการกับข้อมูลที่เป็นพื้นฐานทั่ว ๆ ไป

แต่ผมขอเรียกให้ทราบก่อนว่าถึงแม้จะเป็นโปรแกรมที่สามารถนำไปใช้งานจริงได้ ก็เป็นเพียงโปรแกรมตัวอย่างที่แสดงถึงการใช้งานคอนโทรลเท่านั้น และโปรแกรมในฉบับนี้มีเพียงโปรแกรมเดียว แต่ผมจะเปลี่ยนชื่อไฟล์เป็นระยะ ๆ เพื่อให้ได้ทดลองรัน ทำความเข้าใจเพิ่มในแต่ละขั้นตอนภายหลัง

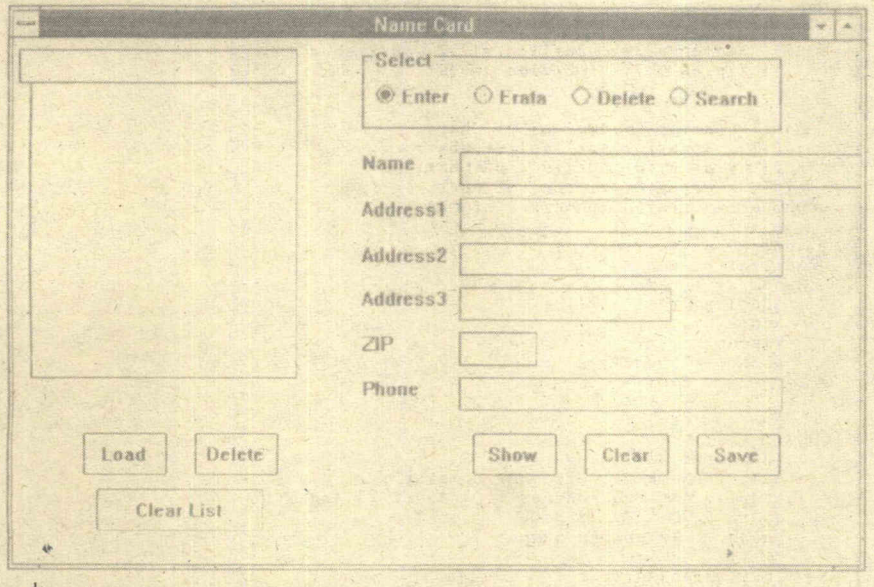
ออกแบบหน้าต่าง

ในการใช้งานคอนโทรลจะให้อยู่ในหน้าต่างแบบใดก็ได้ เพื่อให้โปรแกรมสั้นผมจะให้อยู่ในหน้าต่างหลัก หน้าต่างหลักจึงมีฐานะเป็นหน้าต่างแม่ของพวกคอนโทรลด้วย

เมื่อจะเขียนโปรแกรมก็จะต้องออกแบบโปรแกรมก่อน แต่ผมขอรวบรัดมากล่าวรวมกับการออกแบบหน้าต่างตามรูปที่ 4 (กรุณาอ่านคำอธิบายใต้รูปด้วยครับ) คือ

เมื่อเริ่มต้นรันโปรแกรมจะเซตสถานะการทำงานของโปรแกรมไว้เป็นป้อนข้อมูล (Enter) ซึ่งผู้ใช้อาจเลือกป้อน หรือแก้ (Erata) หรือถอนข้อมูล (Delete) หรือค้นหาข้อมูล (Search) โดยเลือกที่ปุ่มวิทยุในกรอบ Select ตามชื่อภาษาอังกฤษในวงเล็บ

เมื่อเลือกสามสถานะหลังนี้ หากยังไม่ได้โหลดข้อมูลลงรายการให้โหลดด้วยปุ่ม Load ซึ่งจะโหลดและแยกฟิลด์ Name ไปให้แก่กรอบคอมโบกรอบคอมโบจะเป็นผู้นำออกแสดงในกรอบของตน



รูปที่ 4 หน้าต่างของโปรแกรมนามบัตร มีคอนโทรลอยู่ 5 กลุ่มคือ

- กลุ่มที่ 1 คือกรอบ Select และปุ่มวิทยุที่อยู่ในกรอบนี้

- กลุ่มที่ 2 มีกรอบหัวข้อ คือ Name ถึง Phone อยู่ด้านซ้าย และกรอบอีดิตที่กรอบหัวข้อกำกับอยู่ด้านขวา ซึ่งในคำอธิบายรวมเรียกว่า แบบกรอกข้อมูล

- กลุ่มที่ 3 เป็นปุ่มควบคุม Show, Clear และ Save ของแบบกรอกข้อมูล

- กลุ่มที่ 4 คือกรอบคอมโบ

- กลุ่มที่ 5 คือปุ่มควบคุมที่เหลือ เพื่อควบคุมการทำงานของกรอบคอมโบ

ป้อนข้อมูล

ให้ป้อนข้อมูลในแบบกรอกข้อมูล เมื่อจบแต่ละเรคอร์ดคือนามบัตรแต่ละใบ ให้เลือกปุ่ม Save เพื่อเก็บลงไฟล์ และนำฟิลด์ Name ไปให้แก่กรอบรายการเพื่อให้แสดงว่าได้ป้อนข้อมูลใดไปบ้างแล้ว ก่อนป้อนหากมีข้อความค้างอยู่ในแบบกรอบข้อมูลให้เลือกปุ่ม Clear

การค้นหาข้อมูล

การค้นหาข้อมูลดำเนินการในแบบให้ผู้ใช้ค้นด้วยตนเองคือ ค้นจากชื่อในกรอบคอมโบ ซึ่งอาจจะค้นโดยตรง หรือป้อนชื่อที่กรอบป้อนของกรอบคอมโบ ซึ่งอาจจะป้อนด้วยอักษรแรกเพียงตัวเดียว กรอบรายการจะเลื่อนรายการ (ชื่อที่ต้องการอาจไม่อยู่ในกรอบ) ให้รายการแรกที่มีอักษรแรกตรงกันมาอยู่ในกรอบ เพื่อให้ผู้ใช้เลือกโดยตรงต่อไป

เมื่อเลือกได้ชื่อที่ต้องการแล้ว ให้เลือกปุ่ม Show เพื่อนำรายละเอียดออกแสดงในแบบกรอบข้อมูล

แก้ข้อมูล

ให้ค้นหาข้อมูลนำออกแสดงในแบบกรอกข้อมูล แล้วดำเนินการแก้ เมื่อแก้เสร็จให้เลือกปุ่ม Save เพื่อนำไปแก้ในไฟล์

ถอนข้อมูล

ให้ค้นหาข้อมูลซึ่งจะให้แสดงในแบบกรอบข้อมูลหรือไม่ขึ้นอยู่กับผู้ใช้ เมื่อได้ข้อมูลที่ต้องการจะยกเลิก ให้เลือกปุ่ม Delete ที่เป็นปุ่มควบคุม (อยู่ใต้กรอบรายการรอง)

จากการออกแบบ

จะเห็นว่าได้ใช้คอนโทรลต่าง ๆ ตามหน้าที่ของมัน กล่าวคือ

- ได้ใช้ปุ่มวิทยุ ซึ่งจะต้องประกอบด้วยกรอบกลุ่มด้วย ในการกำหนดสถานะการทำงานของโปรแกรมในแต่ละช่วงตอน ซึ่งผู้ใช้จะเปลี่ยนสถานะกลับไปมาอย่างไรก็ได้

- ใช้กรอบอีดิตในการป้อนและแสดงข้อความซึ่งเป็นหน้าที่ของกรอบอีดิตทั้งคู่

- ใช้กรอบหัวข้อแสดงข้อความกำกับกรอบป้อนเพื่อให้ผู้ใช้ทราบว่าจะต้องป้อนข้อความใดที่กรอบใด

- ใช้กรอบคอมโบเพื่อแสดงรายชื่อให้เลือก เพราะคาดว่าจะมีข้อมูลเป็นจำนวนมาก หากใช้กรอบรายการธรรมดาผู้ใช้จะต้องเลือกด้วยตนเองตลอด

- ใช้ปุ่มควบคุมให้มีการดำเนินการทันทีและเสร็จสิ้นการทำงาน ไม่มีสถานะค้างอยู่ดังเช่น ปุ่มวิทยุ หรือปุ่มเซต

ให้กำเนิดคอนโทรล

การออกแบบโปรแกรมยังไม่จบครับ แต่ผมเขียนบทความนี้ในด้านของการใช้งานคอนโทรล เพื่อแสดงว่าเมื่อดำเนินการไปถึงขั้นใดจะได้อะไรบ้าง

การให้กำเนิดคอนโทรล เริ่มจากการอินิต (initialize) คอนโทรลในโปรแกรม Init ของหน้าต่างแม่ หลังจากนี้จะเป็นหน้าที่ของเทอร์โบปาสคาลและวินโดวส์รับไปดำเนินการจนปรากฏภาพคอนโทรลบนจอภาพและใช้งานได้ แต่ก่อนที่จะอีดิตคอนโทรลได้จะต้องเตรียมการในเรื่องต่อไปนี้ก่อน คือ

การดักจับเมสเสจ

คอนโทรลทุกชนิดเว้นกรอบหัวข้อจะส่งเมสเสจเมื่อผู้ใช้ดำเนินการต่อ เช่น คลิ๊ก แต่ในโปรแกรมของเราไม่จำเป็นที่จะต้องดักจับเมสเสจทุกตัว หากต้องการดักจับเมสเสจจากคอนโทรลใดให้กำหนดค่าไอดีให้คอนโทรลนั้น (ซึ่งจะให้ในการอินิต และกำหนดกิจกรรมสำหรับเมสเสจนั้น สำหรับคอนโทรลที่เหลือไม่ต้องกำหนดค่าไอดีให้ คือให้ค่าไอดีเดียวกันหมด เช่น 200 (และจะต้องไม่นำค่านี้ไปใช้ในการกำหนดไอดี) สำหรับโปรแกรมนี้จะพิจารณาได้ดังนี้

- ปุ่มวิทยุทั้งหมดไม่ต้องกำหนดค่าไอดีให้เพราะมีกรอบกลุ่มคือ Select เป็นผู้ส่งเมสเสจแทน คงกำหนดไอดีให้แก่ Select เท่านั้น

- จะต้องกำหนดไอดีให้ปุ่มควบคุมทั้งหมด (และทุกโปรแกรม)

- ไม่ต้องกำหนดไอดีให้กรอบอีดิตทุกกรอบ เพราะมีปุ่ม Save ควบคุมการทำงานอยู่แล้ว

- ไม่ต้องกำหนดไอดีให้กรอบหัวข้อทุกกรอบเพราะไม่ส่งเมสเสจอยู่แล้ว

- ไม่ต้องกำหนดไอดีให้กรอบคอมโบ เพราะมีปุ่ม Show และ Delete ควบคุมการทำงานอยู่แล้ว

สรุปคงกำหนดไอดีให้แก่ Select และปุ่มควบคุมทั้งหมด รวมเป็น 7 ไอดีเท่านั้น แต่เมื่อดูในรูป A1 จะเห็นว่าผมได้กำหนดไอดีให้แก่คอนโทรลเกือบทุกตัว เพราะได้ใช้ทดลองต่าง ๆ ก่อนที่จะมา

```
PROGRAM  NameCard1;   { NMECRD1.PAS }

USES  Strings,WinTypes,WinProcs,WObjects,StdDlgs;

CONST  nmelen = 39;  adr1len = 29;  adr2len = 19;
       ziplen = 7;   phnelen  = 29;

TYPE
  nameT = array[0..nmelen] of Char;
  adr1T = array[0..adr1len] of Char;
  adr2T = array[0..adr2len] of Char;
  zipT  = array[0..ziplen] of Char;
  phoneT = array[0..phnelen] of Char;

  AdrTrnP = ^AdrTrnT;
  AdrTrnT = RECORD
    name    : nameT;
    addr1,addr2 : adr1T;
    addr3   : adr2T;
    zipcde : zipT;
    phone   : phoneT;
            END;


CONST
     dumadr : AdrTrnT =
    (name : 'Name of person or company';
     addr1 : 'Street Address';    addr2 : 'Amphor';
     addr3 : 'City';  zipcde : 'code';
     phone : 'Telephone number');

     nulladr : AdrTrnt =
    (name : ''; addr1 : ''; addr2 : '';  addr3 : '';
     zipcde : ''; phone : '');


CONST
  id_nme   = 201;    id_adr1   = 202;      id_adr2  = 203;
  id_adr3  = 204;    id_zip    = 205;      id_phne  = 206;
  id_combo = 207;    id_select = 208;      id_entr  = 209;
  id_erata = 210;    id_del    = 211;      id_srch  = 212;
  id_load  = 213;    id_delete = 214;      id_shwd  = 215;
  id_clr   = 216;    id_save   = 217;      id_clrlist = 218;
```

รูป A1

```
dum := New(PEdit, Init(@self,id_name,'',
                       100,30,200,54,20,false));
  ............
  1  dum^.Clear;

  2  dum := PEdit(ChildWithID(id_name));
     dum^.Clear;
```

รูปที่ 5
แสดงการใช้งานคอนโทรลเมื่อ
1. เก็บ dum ไว้
2. ไม่ได้เก็บ (เป็นตัวแปรตัวใหม่)

เขียนโปรแกรมนี้แล้วไม่ได้ลบออกครับ

ในรูปที่ 4 มีปุ่มควบคุมอีก 1 ปุ่ม คือ Clear List ที่ยังไม่ได้กล่าวถึง เพื่อให้ลบค่าในลิสต์ในกรอบคอมโบ ซึ่งจะทำให้ลบภาพในกรอบคอมโบด้วย ปุ่มนี้เพื่อการทดลองเท่านั้น ในการใช้งานจริงไม่ต้องมี รวมทั้งปุ่ม Load ด้วย โดยเขียนโปรแกรมให้โหลดโดยอัตโนมัติ

การส่งสื่อความ

การที่เราจะส่งสื่อความไปให้ออปเจ็กต์ได้ จะต้องมีอินสแตนซ์หรือพอยน์เตอร์ไปยังออปเจ็กต์นั้น เช่นตามรูปที่ 5 ได้อีดิตกรอบอีดิต โดยให้ค่าพอยน์เตอร์แก่ dum ทำให้สามารถส่งสื่อความ Clear ไปให้ออปเจ็กต์นี้ได้

และเพราะเราอีดิตคอนโทรลในหน้าต่างแม่ หน้าต่างแม่จะเก็บพอยน์เตอร์ไปยังออปเจ็กต์คือ ค่าเดียวกับใน dum ไว้ในลิงค์ลิสต์ (childlist) ของตน ซึ่งสามารถอ่านมาใช้งานได้ดังโปรแกรมท่อนสุดท้ายในรูปที่ 5 นั่นคือเราไม่ต้องเก็บค่า dum ไว้เช่นให้เป็นตัวแปรโลคอล

แต่เทอร์โบปาสคาลแนะนำว่า หากต้องการส่งสื่อความไปยังออปเจ็กต์ใดให้เก็บพอยน์เตอร์ของออปเจ็กต์นั้นไว้ โดยกำหนดให้เป็นฟิลด์ข้อมูลฟิลด์หนึ่งของสกุล เรียกว่าฟิลด์ออปเจ็กต์ (object field)

ดังนั้นจึงต้องพิจารณาว่าจะต้องกำหนดฟิลด์ออปเจ็กต์ให้แก่คอนโทรลใดบ้าง ซึ่งผมจะพิจารณาในเรื่องที่ง่ายก่อน คือ

- ปุ่มควบคุมทั้งหมด และกรอบหัวข้อทั้งหมดไม่ต้องจัดฟิลด์ออปเจ็กต์ให้ เพราะไม่มีสิ่งใดจะให้ดำเนินการ (เช่นให้แก้ข้อความที่เป็นชื่อ)

- กรอบอีดิตทั้งหมด ไม่ต้องจัดฟิลด์ออปเจ็กต์ให้เพราะใช้ Save ควบคุมการทำงาน จึงไม่มีสิ่งใดจะติดต่อด้วย

- กรอบคอมโบ จะต้องจัดฟิลด์ออปเจ็กต์ เพราะจะต้องสอบถามว่าผู้ใช้เลือกรายการใด และขอค่าที่เลือก

- กรอบกลุ่มคือ Select ไม่ต้องการส่งสื่อความไปให้ไม่ต้องจัดฟิลด์ออปเจ็กต์ให้ (แต่ผมเผลอกำหนดให้)

- ปุ่มวิทยุ เมื่อ Select ส่งเมสเสจมาว่าผู้ใช้เลือกปุ่มวิทยุ (ปุ่มใดปุ่มหนึ่ง) เราจะต้องตรวจสอบปุ่มวิทยุแต่ละปุ่มว่าปุ่มใดที่ถูกเลือก ซึ่งผมเห็นว่าหากทำเป็นอะเรย์ของออปเจ็กต์จะเขียนโปรแกรมได้สั้นกว่า

สรุปจะต้องกำหนดฟิลด์ออปเจ็กต์ 2 ฟิลด์และฟิลด์อะเรย์ของออปเจ็กต์อีก 1 ฟิลด์ นำไปกำหนดแบบข้อมูล และกำหนดสกุล WndProC ได้ตามรูป A2 โดยที่ comboB คือฟิลด์ออปเจ็กต์ของกรอบคอมโบ selectB ของกรอบกลุ่ม และ rbArray เป็นอะเรย์ของออปเจ็กต์ปุ่มวิทยุ สำหรับฟิลด์ select เพื่อเก็บสถานะที่ได้จากการตรวจสอบปุ่มวิทยุ

```
TYPE
  DemoApplC = object(TApplication)
    procedure  InitMainWindow; virtual;
               end;

TYPE
  RadioT = (rentr,rerata,rdel,rsrch);
  rbArrayT = ARRAY[RadioT] OF PRadioButton;

TYPE
  WndProCP = ^WndProC;
  WndProC  = object(TWindow)
    select : RadioT;
    tran   : AdrTrnT;

    comboB  : PComboBox;
    selectB : PGroupBox;
    rbArray : rbArrayT;

  constructor  Init(apar : PWindowsObject; atitle : PChar);
  procedure  SetupWindow;  virtual;
               end;


(* ---------------------- WndProC Routine ---------------------- *)

CONST
  sname : array[0..5] of PChar =
    ('Name','Address1','Address2','Address3','ZIP','Phone');

  butname : array[0..4] of PChar =
    ('Load','Delete','Show','Clear','Save');

  edlen : array[0..5] of Integer =
    (nmelen,adr1len,adr1len,adr2len,ziplen,phnelen);
```

รูป A2

การให้ค่าเริ่มต้น

เมื่อมีปุ่มวิทยุ ซึ่งอาจถือได้ว่าเป็นปุ่มสถานะจึงต้องมีสถานะเริ่มต้น โดยที่ผมกำหนดสถานะเริ่มต้นเป็นป้อนข้อมูล นั่นคือผมจะต้องให้ปุ่ม Enter (ตามรูปที่ 4) มีสถานะเป็นเซต

โดยที่เมื่อเราอีดิตหน้าต่าง (ทุกชนิดรวมทั้งคอนโทรลด้วย) จะเป็นการให้กำเนิดออปเจ็กต์ทางซีกของเทอร์โบปาสคาลและยังไม่สมบูรณ์ด้วย ส่วนหน้าต่างตัวจริงซึ่งเป็นซีกของวินโดวส์ยังไม่เกิดขึ้น จนกว่าจะได้ทำงานใน SetUpWindow ซึ่งจะเรียกใช้ Create ให้กำเนิดหน้าต่างตัวจริง (เทอร์โบปาสคาลเปรียบเทียบออปเจ็กต์หน้าต่างว่าเทียบได้กับไฟล์ในภาษาปาสคาล ส่วนหน้าต่างตัวจริงเทียบได้กับไฟล์ในดิสก์)

เมื่อเป็นเช่นนี้ในโปรแกรมของกิจกรรม Init ของหน้าต่างแม่ ถึงแม้จะได้ Init หน้าต่างลูกแล้วก็ยังไม่สามารถส่งสื่อความไปให้หน้าต่างลูกได้ (เว้นบางสื่อความ) โอกาสแรกที่จะส่งสื่อความไปให้แก่หน้าต่างลูกคือใน SetUpWindow ของหน้าต่างแม่และหลังจากเรียกใช้ SetUpWindow ของสกุลบรรพบุรุษของหน้าต่างแม่แล้ว ซึ่งจะเกิดก่อนการแสดงภาพหน้าต่าง (ทั้งหมดและทุกชนิด) บนจอภาพ

นั่นคือ หากต้องการให้ค่าเริ่มต้นแก่คอนโทรล เช่น ให้ค่ารายการแก่ลิสต์ และสำหรับโปรแกรมนี้คือการเซตค่าแก่ปุ่ม Enter (คือ Check) ให้ทำใน SetUpWindow

ให้หน้าต่างอ่านเขียนกับคอนโทรล

คอนโทรลทุกชนิดมีข้อมูลอย่างน้อยที่สุดคือข้อความที่ให้เมื่อ Init ดังที่เห็นเป็นชื่อของคอนโทรลต่าง ๆ เราสามารถอ่านเขียนกับข้อมูลเหล่านี้ได้ และทำได้หลายวิธีมีรายละเอียดซึ่งผมจะขอนำไปกล่าวในหัวข้อ "อ่านเขียนข้อมูลกับคอนโทรล" (ในฉบับต่อไป)

แต่ในขั้นตอนนี้ผมจำเป็นที่จะต้องนำวิธีการหนึ่งคือ การให้หน้าต่างอ่านเขียนข้อมูลกับคอนโทรลมากล่าวถึงก่อนเพื่อจะรวมเรื่องที่จะต้องดำเนินการเมื่ออีนิตไว้ในที่เดียวกัน

มีเรื่องสำคัญที่จะต้องกล่าวถึงก่อนอีกครับ คือ การอ่านเขียนกับคอนโทรลเป็นการลอกค่าคือ เมื่อเราเขียนเป็นการให้ค่าแก่คอนโทรล คอนโทรลจะลอกข้อมูลที่ให้ไปเก็บไว้ในตัวมัน (คือไม่ได้เก็บค่าพอยน์เตอร์) และเมื่อเราอ่านข้อมูลจากคอนโทรล คอนโทรลจะลอกค่าให้เช่นกัน ดังนั้นตัวแปรที่ส่งไปรับค่าจะต้องมีหน่วยความจำรองรับ ที่กล่าวเช่นนี้เพราะได้กำหนดแบบข้อมูลไว้เป็น PChar อาจทำให้สับสนได้

เมื่อต้องการให้หน้าต่างอ่านเขียนกับคอนโทรลให้ จะต้องดำเนินการดังนี้

1. กำหนดว่าจะให้อ่านเขียนกับคอนโทรลใดบ้าง ตามตัวอย่างนี้คือกรอบอีนิตทั้งหมดและเท่านั้น

2. กำหนดตัวแปรเรคอร์ดให้มีฟิลด์ตรงตามคอนโทรลที่อ่านเขียนด้วย ดังการกำหนดแบบข้อมูล AdrTrnT ในรูป A1 และนำไปกำหนดฟิลด์ tran ในรูป A2

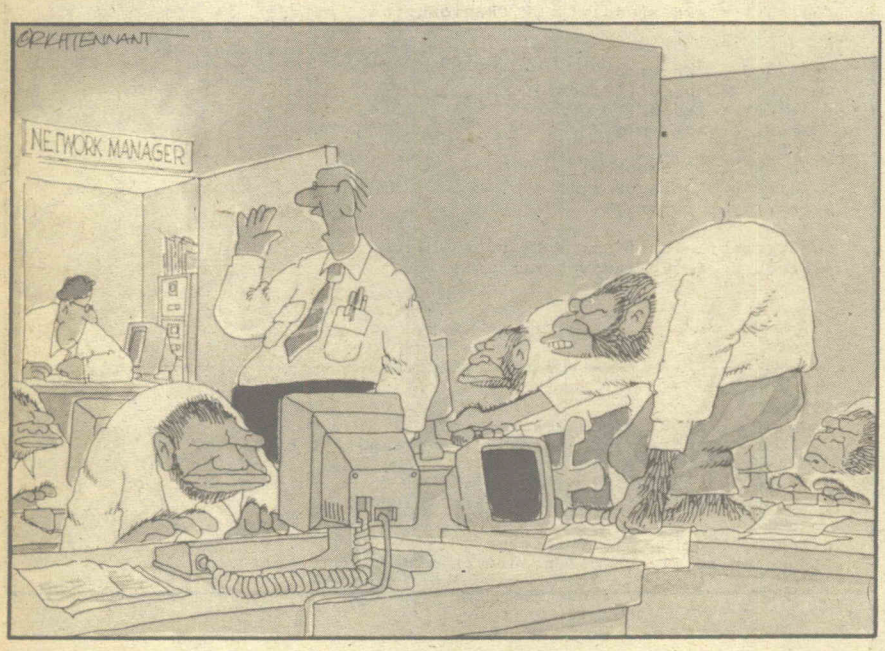
3. ให้ค่าแอดเดรสของเรคอร์ดนี้แก่ฟิลด์ TransferBuffer ของหน้าต่างแม่

4. ในการอีนิตคอนโทรลจะต้องอีนิตคอนโทรลที่จะให้อ่านเขียนด้วยตามลำดับฟิลด์ของเรคอร์ด

5. จะต้องอีนาเบิ้ลคอนโทรลเหล่านี้เพื่อให้รับการผ่านค่ากับหน้าต่างแม่ด้วย EnableTransfer

อีนิต

ตอนนี้เราก็พร้อมที่จะให้กำเนิดคอนโทรล คือพร้อมที่จะเขียนโปรแกรมสำหรับช่วงตอนนี้ ดังที่ได้กำหนดสกุล



ของหน้าต่างแม่คือ WndProC ซึ่งเป็นหน้าต่างหลักด้วยไว้ในรูป A2 และจะดำเนินการเพียงเท่านี้ก่อน (มีส่วนของการกำหนดค่าคงตัวซึ่งควรจะอยู่ในรูป A3 อยู่ในรูป A2 ด้วย เพื่อไม่ให้รูป A3 ยาวเกินไป)

การอีนิตหน้าต่างแม่และคอนโทรลอยู่ใน Init ของหน้าต่างแม่ คือในรูป A3 มีการดำเนินการดังนี้

1. อีนิตหน้าต่างแม่ แล้วให้ค่าขนาดของหน้าต่างใหม่ (ไม่ใช้ค่าที่เตรียมไว้) เพื่อให้ได้ขนาดที่ต้องการ

2. อีนิตกรอบอีดิตทั้ง 6 กรอบในวงรอบ FOR ค่า @self ที่ให้หมายถึงพอยน์เตอร์ไปยังออปเจ็กต์แม่ ตามด้วยค่าไอดี ค่า x, y (โคออร์ดิเนตของจุดมุมบนซ้ายของคอนโทรลในหน้าต่างแม่) ค่า w และ h (ความกว้างและสูงของคอนโทรล) ค่า lem คือความยาวของข้อความที่ให้ป้อนได้ (ยาวกว่าความยาวของกรอบได้) ค่า false คือให้เป็นเอดิเตอร์บรรทัดเดียว

3. ในวงรอบ FOR เช่นกันได้ EnableTransfer ให้กรอบอีดิตทุกกรอบเป็นข้อยกเว้นที่ให้ใช้สื่อความนี้ในโปรแกรมอีนิตของหน้าต่างแม่ได้

4. อีนิตกรอบคอมโบ กรอบกลุ่ม และปุ่มวิทยุ ในอีนิตของปุ่มวิทยุได้ให้ค่าพอยน์เตอร์ไปยังออปเจ็กต์กรอบกลุ่มด้วยคือ selectB

5. อีนิตปุ่มควบคุม 5 ปุ่มในวงรอบ FOR แล้วอีนิตปุ่ม Clear List เป็นพิเศษอีก 1 ปุ่ม

6. อีนิตกรอบหัวข้อทั้ง 6 กรอบในวงรอบ FOR

7. ได้ให้แอดเดรสของเรคอร์ดที่เตรียมไว้คือ tran แก่ฟิลด์ TransferBuffer ของหน้าต่างแม่

```
CONSTRUCTOR  WndProC.Init(apar : PWindowsObject; atitle : PChar);
CONST  col1 = 4;  col2 = 250;  row2 = 80; col3 = 320;  row3 = 80;
       col4 = 250;  col5 : Integer = 50;  row5 = 280;  ydif = 32;
VAR    edB : PEdit;  statB : PStatic;   botB : PButton;
       id,len,wid : Integer;
BEGIN
  TWindow.Init(apar,atitle);
  attr.x := 10;  attr.y := 30;  attr.w := 620;  attr.h := 400;

  for  id := 0 to 5  do
    begin
    len := edlen[id];  wid := len * 8;  inc(len);
    if  wid > 290  then  wid := 290;
    edB := New(PEdit, Init(@self,id_nme+id,'',
                           col3,row3+ydif*id,wid,24,len,false));
    edB^.EnableTransfer;
    end;

  comboB := New(PComboBox, Init(@self,id_combo,col1,8,200,240,
                                cbs_Simple,nmelen));
  selectB := New(PGroupBox, Init(@self,id_select,'Select',
                                 col4,4,300,60));
  rbArray[rentr] := New(PRadioButton, Init(@self,id_entr,'Enter',
                                           col4+10,30,60,20,selectB));
  rbArray[rerata] := New(PRadioButton, Init(@self,id_erata,'Erata',
                                            col4+80,30,60,20,selectB));
  rbArray[rdel] := New(PRadioButton, Init(@self,id_del,'Delete',
                                          col4+150,30,60,20,selectB));
  rbArray[rsrch] := New(PRadioButton, Init(@self,id_srch,'Search',
                                           col4+220,30,65,20,selectB));
  for  id := 0 to 4  do
    begin
    if  id = 2  then  col5 := col5 + 120;
    botB := New(PButton, Init(@self,id_load+id,butname[id],
                              col5+80*id,row5,60,30,false));
    end;
  botB := New(PButton, Init(@self,id_clrlist,'Clear List',
                            60,row5+40,120,30,false));
  for  id := 0 to 5  do
    statB := New(PStatic, Init(@self,200,sname[id],
                               col2,row2+ydif*id,70,24,8));
  tran := nulladr;   TransferBuffer := @tran;  EnableKBHandler;
END;
```

รูป A3

```
PROCEDURE WndProC.SetupWindow;
BEGIN
  TWindow.SetupWindow;
  rbArray[rentr]^.Check;   select := rentr;
END;


(* ==================== DemoApp1C Routine ========================== *)

PROCEDURE  DemoApp1C.InitMainWindow;
BEGIN
  MainWindow := New(WndProCP, Init(nil,'Name Card'));
END;



(*  ----------------------  MAIN PROGRAM  ------------------- *)


VAR   nmecrd : DemoApp1C;

BEGIN
  nmecrd.Init('Address');
  nmecrd.Run;
  nmecrd.Done;
END.
```

รูป A4

SetupWindow

ใน SetupWindow รูป A4 จะต้องเรียกใช้กิจกรรมนี้ของบรรพบุรุษเพื่อให้กำเนิดหน้าต่างทุกชนิดก่อน หลังจากนี้จึงจะส่งสื่อความไปให้หน้าต่างได้ และสำหรับโปรแกรมนี้ได้รีเซตปุ่ม Enter โดยส่งสื่อความ Check ไปให้กับให้เก็บค่าสถานะไว้ใน select

ทดสอบโปรแกรม

ขอให้ป้อนโปรแกรมทั้งหมดนี้และส่วนที่เหลือในรูป A4 ด้วยเก็บในไฟล์ NMECRD1.PAS แล้วทดลองรันครับ จะได้ผลการรันตามรูป A4 ซึ่งคุณสามารถทดลองในสิ่งต่อไปนี้ได้

- คลิ๊กที่ปุ่มวิทยุ เมื่อคลิ๊กที่ปุ่มใดปุ่มนั้นจะถูกเซตแสดงเป็นจุดดำในวงกลม และปุ่มอื่นที่เซตไว้เดิมจะถูกรีเซตคือ จุดดำหายไป
- ป้อนข้อมูลที่กรอบอีดิตได้ทุกกรอบ รวมทั้งของกรอบคอมโบด้วย
- เมื่อคลิ๊กที่ปุ่มควบคุมปุ่มนั้นจะยุบลง (ถ้าคลิ๊กค้างไว้จะเห็นถนัด)

จะเห็นว่าวินโดวส์ได้เตรียมการต่าง ๆ ไว้ให้เรามากพอสมควร ยังขาดอยู่แต่โปรแกรมของเราที่จะติดต่อกับคอนโทรลต่าง ๆ

แต่ก็มีปัญหาที่ผมยังไม่ทราบว่าเป็นเพราะเหตุใดคือ เมื่อเริ่มรันยังใช้คีย์ tab ไม่ได้ต้องใช้เม้าส์ก่อน ตัวอย่างของเทอร์โบปาสคาลก็ไม่มีที่ให้ใช้คีย์-บอร์ด

สำหรับไฟล์ NMECRD1.PAS ผมเก็บไว้สำหรับการศึกษา โดยก๊อบปี้ให้ไฟล์ NMECRD2.PAS แล้วเปลี่ยนชื่อโปรแกรมเป็น NameCard2 เพื่อต่อเติมโปรแกรมต่อไป

ออกแบบต่อ

โปรแกรม NameCard1 เพื่อแสดงว่าจะให้กำเนิดคอนโทรลอย่างไร และวินโดวส์ได้เตรียมสิ่งใดไว้บ้างแล้ว โปรแกรมต่อไปนี้จะแสดงว่าเราจะใช้งานคอนโทรลได้อย่างไร ซึ่งจะต้องมีการเตรียมการในเรื่องอื่นด้วย กล่าวคือ

ไฟล์เทียม

จากการออกแบบจะเก็บข้อมูลไว้ในไฟล์ ซึ่งอาจจะใช้ไฟล์ตามแบบของปาสคาลก็ได้ แต่ในเทอร์โบปาสคาลได้กำหนดสกุลในชุดของ TStream คือ สตรีมไว้ เช่น TDosStream ก็คือไฟล์ในดิสค์ ซึ่งผมควรจะนำมาใช้แต่ผมต้องการให้โปรแกรมในขั้นตอนนี้เป็นเรื่องของการใช้คอนโทรล จึงขอพักเรื่องของไฟล์ไว้ก่อน

โดยจะเก็บข้อมูลที่ผู้ใช้ป้อนไว้ในที่เก็บข้อมูล (ถึงแม้จะใช้สตรีมแต่วิธีที่สะดวกคือ เก็บข้อมูลไว้ในที่เก็บข้อมูลก่อน แล้วเก็บลงสตรีมภายหลัง) สำหรับที่เก็บข้อมูลในตอนก่อนผมได้นำสกุล TCollection มาใช้ แต่ในตอนนี้ผมต้องการที่เก็บข้อมูลที่จัดเรียงลำดับข้อมูลด้วยคือสกุล TSortedCollection

ทั้งนี้เพื่อให้การจัดเก็บข้อมูลในที่เก็บข้อมูล กับในลิสต์ของกรอบคอมโบเรียงตรงกัน เพื่อใช้เมื่อผู้ใช้เลือกรายชื่อในลิสต์ ผมจะนำลำดับที่ในลิสต์ไปอ่านข้อมูล (ทั้งเรคอร์ด) ในที่เก็บข้อมูลได้โดยตรงไม่ต้องไปค้นหาอีก

```
  AddrCollectCP = ^AddrCollectC;
  AddrCollectC = OBJECT(TSortedCollection)
    constructor  Init(alimit,adelta : Integer);
    function  KeyOf(item : Pointer) : Pointer; virtual;
    function  Compare(key1,key2 : Pointer) : Integer; virtual;
              END;


CONSTRUCTOR  AddrCollectC.Init(alimit,adelta : Integer);
BEGIN   TsortedCollection.Init(alimit,adelta)  END;


FUNCTION  AddrCollectC.KeyOf(item : Pointer) : Pointer;
BEGIN
  KeyOf := @AddrCP(item)^.adrrcrd.name
END;


FUNCTION  AddrCollectC.Compare(key1,key2 : Pointer) : Integer;
VAR    keya,keyb : PChar;  res : Integer;
BEGIN
  Compare := StrIComp(PChar(key1), PChar(key2))
END;
```

รูป B1

```
PROCEDURE WndProC.SetupWindow;
BEGIN
  TWindow.SetupWindow;
  fakefi := New(AddrCollectCP. Init(100,10));
  rbArray[rentr]^.Check;   select := rentr;
END;


DESTRUCTOR  WndProC.Done;
BEGIN
  Dispose(fakefi,Done);
  TWindow.Done;
END;
```

รูป B2

```
AddrCP = ^AddrC;
AddrC  = OBJECT(TObject)
  adrrcrd  : AdrTrnT;
constructor  Init(drcrd : AdrTrnT);
procedure  SetData(drcrd : AdrTrnT);
procedure  GetData(var drcrd : AdrTrnT);
function   GetName : Pointer;
         END;
```

รูป B3

สำหรับที่เก็บข้อมูลต่อไปผมจะขอเรียกเป็นไฟล์เทียมเมื่อกล่าวถึงในฐานะของไฟล์ เพื่อให้สอดคล้องกับการเซฟ (ปุ่ม Save) และการโหลด (ปุ่ม Load) จึงจะดำเนินการกับไฟล์เทียมนี้

การใช้สกุล TSortedCollection จะต้องปรับปรุงก่อน โดยกำหนดสกุลใหม่ดังในรูป B1 ซึ่งจะต้องมีชื่อของกิจกรรมตามนี้ กล่าวคือ

- KeyOf เป็นฟังก์ชันให้ค่าแอดเดรสของฟิลด์ที่เราจะใช้เป็นคีย์ ตามโปรแกรมคือฟิลด์ name (ชื่อ) ส่วนแบบข้อมูลของ addrrerd จะกล่าวในหัวข้อต่อไป

- Compare เป็นฟังก์ชันเทียบค่า ซึ่งจะต้องเขียนโปรแกรมให้ค่า เป็น -1 เมื่อค่าของ Key1 น้อยกว่า Key2 เป็น 0 เมื่อเท่ากันเป็น 1 เมื่อมากกว่า ซึ่งถ้าเป็นสตริงใช้ StrIComp จะให้ค่าตามนี้ สำหรับ StrIComp จะเทียบสตริงโดยไม่แยกอักษรใหญ่-เล็ก

ขอให้ป้อนโปรแกรมตามรูป B1 ไว้ในตอนต้น ๆ ของโปรแกรม และขอให้แก้โปรแกรมของกิจกรรม SetupWindow ในรูป A4 เดิม ให้เป็นดังรูป B2 คืออีนิต fakefi คือไฟล์เทียมให้เป็นที่เก็บข้อมูลขนาด 100 ช่อง และให้ขยายได้ครั้งละ 10 ช่อง ด้วยการจองหน่วยความจำ และยกเลิกใน Done รูปเดียวกันนี้

ออปเจ็กต์ของข้อมูล

เมื่อจะเก็บข้อมูลในที่เก็บข้อมูล จะต้องกำหนดสกุลให้ดังการกำหนดสกุล AddrC ในรูป B3 โดยที่ผมเก็บข้อมูลไว้ทั้งเรคอร์ด เพราะการดำเนินกรรมวิธีส่วนมากดำเนินการเป็นเรคอร์ด และได้เพิ่มกิจกรรมตามที่เห็นในรูป มีโปรแกรมอยู่ในรูป B4 ขอให้ป้อนโปรแกรมตามรูปทั้งสองนี้ไว้ตอนต้นของโปรแกรมเช่นกัน

ขออภัย

ในไมโคร ฯ 84 ผมได้กล่าวถึงการกำหนดสกุลให้แก่ข้อมูลคือสกุล AgeC โดยไม่มี constructor ก็ต้องขออภัยครับ สกุลของข้อมูลที่จะเก็บในที่เก็บข้อมูลจะต้องมี constructor เสมอ ทั้งนี้เพื่อ

```
CONSTRUCTOR  AddrC.Init(drcrd : AdrTrnT);
BEGIN
  adrrcrd := drcrd;
END;

PROCEDURE  AddrC.SetData(drcrd : AdrTrnT);
BEGIN   adrrcrd := drcrd   END;

PROCEDURE  AddrC.GetData(var drcrd : AdrTrnT);
BEGIN   drcrd := adrrcrd          END;

FUNCTION  AddrC.GetName : Pointer;
BEGIN   GetName := @adrrcrd.name  END;
```

รูป B4

```
TYPE
  RadioT = (rentr,rerata,rdel,rsrch);
  rbArrayT = ARRAY[RadioT] OF PRadioButton;

TYPE
  WndProCP = ^WndProC;
  WndProC  = object(TWindow)
    fakefi  : AddrCollectCP;
    select : RadioT;
    tran   : AdrTrnT;

    comboB  : PComboBox;
    selectB : PGroupBox;
    rbArray : rbArrayT;

  constructor  Init(apar : PWindowsObject; atitle : PChar);
  destructor   Done;  virtual;
  procedure  SetupWindow;  virtual;
  procedure  idSelect(var msg : TMessage); virtual id_first + id_Select;
  procedure  idShwd(var msg : TMessage); virtual id_first + id_shwd;
  procedure  idClr(var msg : TMessage); virtual id_first + id_clr;
  procedure  idSave(var msg : TMessage); virtual id_first + id_save;
  procedure  idLoad(var msg : TMessage); virtual id_first + id_load;
  procedure  idDelete(var msg : TMessage); virtual id_first + id_delete;
  procedure  idClrList(var msg : TMessage); virtual id_first + id_clrlist;
             end;
```

รูป B5

```
PROCEDURE  WndProC.idLoad(var msg : TMessage);
VAR   adr : AddrCP;  idx : Integer;
BEGIN
  comboB^.ClearList;
  for  idx := 0 to fakefi^.Count-1  do
    begin  adr := fakefi^.At(idx);
           comboB^.InsertString(adr^.GetName,-1)   end;
END;
```

รูป B6

ที่เก็บข้อมูลจะยกเลิกข้อมูลในที่เก็บ จะได้ใช้ตาราง VMT ของสกุลใหม่ หากไม่มีจะใช้ของสกุลแม่ได้ขนาดของออปเจ็กต์ไม่ถูกต้อง เพราะในสกุลใหม่มีฟิลด์ข้อมูลมากกว่า

กิจกรรมรับเมสเสจ

เพื่อจะดักจับเมสเสจดังกล่าวมาจะต้องเตรียมกิจกรรมไว้ดังแสดงในรูป B5 (ขอให้เพิ่มในสกุล WndProC รูป A2 ให้เป็นตามรูป B5 ด้วย) กิจกรรมที่เพิ่มคือ Done กับกิจกรรมรับเมสเสจคือที่มีชื่อขึ้นต้นด้วย id

เขียนโปรแกรม

ตั้งแต่เขียนโปรแกรมบนวินโดวส์มาผมเพิ่งรู้สึกว่าได้เขียนโปรแกรมจริงจังในตอนนี้ การเขียนโปรแกรมใช้งานคอนโทรลก็คือ การเขียนโปรแกรมของกิจกรรมที่จัดไว้รับเมสเสจหรือจะคิดเป็นดักจับได้เช่นกัน โดยเขียนให้เป็นไปตามความต้องการของเรา รวมทั้งการส่งสื่อความไปให้คอนโทรลด้วย โปรแกรมจึงเป็นท่อน ๆ ไม่ต่อเนื่องกัน สำหรับโปรแกรมนามบัตร ได้แก่

เซตสถานะ

เมื่อผู้ใช้เลือกปุ่มวิทยุปุ่มใดปุ่มหนึ่ง กรอบ Select (ความจริงคือวินโดวส์) จะส่งสื่อความ id_Select มาให้แก่หน้าต่าง ซึ่งเราได้กำหนดกิจกรรม id-Select ไว้รับแล้ว

หน้าที่ของเราต่อไปคือเขียนโปรแกรมให้ตามรูป B6 โดยส่งสื่อความ GetCheck ไปถามและขอค่าสถานะถ้าปุ่มถูกเซตจะมีสถานะเป็น check คือ 1 จะให้ select เก็บค่าไว้ตรวจสอบว่าปุ่มใดเซต ซึ่งจะมีเพียงปุ่มเดียว

โหลดข้อมูลให้ลิสต์

เช่นกันเมื่อผู้ใช้เลือกปุ่มโหลดก็จะเป็นเรื่องของกิจกรรม idLoad มีโปรแกรมอยู่ในรูป B6 โดย

- ยกเลิกรายการในลิสต์ทั้งหมด
- อ่านข้อมูลทั้งหมดจากไฟล์เทียมทีละ 1 เรคอร์ด นำฟิลด์ name (อ่านด้วย GetName) ไปเก็บในลิสต์ด้วยสื่อความ String แต่ค่า -1 ที่ให้หมายถึงให้เก็บต่อท้ายทำให้เรียงลำดับตามไฟล์เทียมซึ่งเรียงลำดับตามอักษร

เก็บข้อมูลลงไฟล์

เมื่อผู้ใช้เลือกปุ่ม Save กิจกรรม idSave จะเป็นผู้รับเมสเสจ มีโปรแกรมอยู่ในรูป B7 ซึ่งให้ดำเนินการดังนี้

- ตรวจว่าถ้าเป็นการป้อนหรือแก้จึงจะดำเนินการต่อ
- ให้หน้าต่างถ่ายทอดข้อมูลจากแบบกรอกข้อมูลคือกรอบอีดิตทุกกรอบให้แก่ tran
- ตรวจฟิลด์ name ถ้าว่างจะไม่ดำเนินการต่อเพราะจะเป็นคีย์

```
PROCEDURE  WndProC.idSave(var msg : TMessage);
VAR   idx : Integer;  rcrd : AddrCP;
BEGIN
  if  (select <> rentr) and (select <> rerata)  then  Exit;
  TransferData(tf_GetData);
  if  tran.name = ''  then  Exit;
  if  select = rentr  then
    begin  rcrd := New(AddrCP, Init(tran));
           fakefi^.Insert(rcrd);  idx := fakefi^.IndexOf(rcrd);
           comboB^.InsertString(tran.name,-1);  end
  else
    begin  idx := comboB^.GetSelIndex;
           fakefi^.AtPut(idx, New(AddrCP, Init(tran)))  end;
  tran := nulladr;  TransferData(tf_SetData);
END;
```

รูป B7

```
PROCEDURE  WndProC.idShwd(var msg : TMessage);
VAR   tem : AddrCP;  idx : Integer;
BEGIN
  if  select = rentr  then  Exit;
  idx := comboB^.GetSelIndex;
  if  idx < 0  then  Exit;
  tem := fakefi^.At(idx);  tem^.GetData(tran);
  TransferData(tf_SetData);
END;

PROCEDURE  WndProC.idClr(var msg : TMessage);
BEGIN
  tran := nulladr;  TransferData(tf_SetData);
END;

PROCEDURE  WndProC.idDelete(var msg : TMessage);
VAR   idx : Integer;
BEGIN
  if  select <> rdel  then  Exit;
  idx := comboB^.GetSelIndex;
  if  idx >= 0  then
    begin  comboB^.DeleteString(idx);
           fakefi^.AtDelete(idx)  end;
END;

PROCEDURE  WndProC.idClrList(var msg : TMessage);
BEGIN   comboB^.ClearList    END;
```

รูป B8

- ถ้าเป็นการป้อนจะนำไปเก็บในไฟล์เทียม และนำไปเก็บในลิสต์ในแบบเก็บต่อท้าย เพื่อให้ผู้ใช้เห็นว่าได้ป้อนข้อมูลใดไปบ้าง

- ถ้าเป็นการแก้ จะส่งสื่อความไปขอตำแหน่งในลิสต์จากกรอบ comboB เพราะผู้ใช้จะต้องเลือกรายการชื่อก่อนจึงจะแก้ได้ นำตำแหน่งนี้ไปแก้ในไฟล์เทียม

- ให้ลบข้อมูลใน tran ด้วย null-adr

แสดงข้อมูล

เมื่อผู้ใช้เลือกชื่อในลิสต์แล้วจะเลือกปุ่ม Show เพื่อให้แสดงข้อมูลทั้งเรคอร์ดกิจกรรม idShwd จะเป็นผู้รับเมสเสจ โปรแกรมอยู่ในรูป B8 คือ

- ตรวจสอบว่าถ้าเป็นการป้อนไม่ดำเนินการต่อ

- ส่งสื่อความไปขอตำแหน่งที่จาก comboB

- นำไปอ่านออปเจ็กต์จากไฟล์เทียม เพื่อให้ค่าข้อมูลแก่ tran

- ให้หน้าต่างถ่ายค่าให้แก่แบบกรอกข้อมูลเท่ากับให้นำออกแสดง

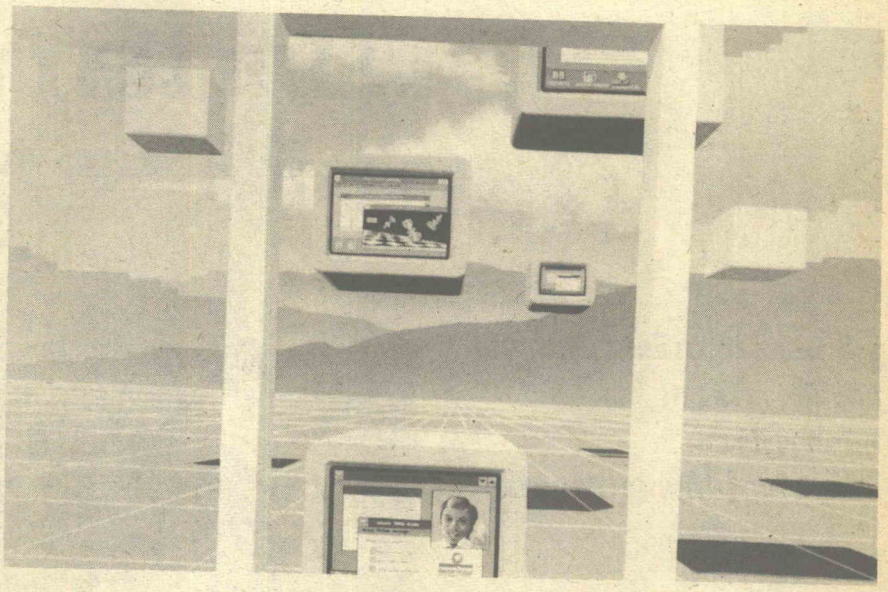
ถอนข้อมูลจากไฟล์

เมื่อผู้ใช้ต้องการถอนข้อมูลจะเลือกชื่อในลิสต์ก่อนแล้วเลือกปุ่ม Delete อันเป็นหน้าที่ของกิจกรรม idDelete ตามโปรแกรมในรูป B8 คือ

- ถ้าผู้ใช้ไม่เซตสถานะไว้เป็นถอนข้อมูลจะไม่ดำเนินการต่อ

- ส่งสื่อความไปถามกรอบ comboB ว่าผู้ใช้เลือกรายการใด ให้ถอนรายการนั้นจากลิสต์

- แล้วนำตำแหน่งไปถอนข้อมูลจากไฟล์เทียม



ยกเลิกข้อมูลในลิสต์

เมื่อผู้ใช้ต้องการยกเลิกรายการในลิสต์ทั้งหมดจะเลือกปุ่ม Clear List มาทำงานในโปรแกรมของกิจกรรม idClrList ในรูป B8 ซึ่งเพียงแต่ส่งสื่อความ ClearList ไปให้ comboB จะเป็นการลบภาพในกรอบรายการด้วย

ลบแบบบกรอบข้อมูล

เมื่อผู้ใช้ต้องการลบแบบกรอกข้อมูลจะเลือกปุ่ม Clear มาทำงานที่โปรแกรมของกิจกรรม idClr รูป B8 ซึ่งเพียงแต่ให้ค่าเรคอร์ดที่มีแต่สตริงว่างแก่แบบกรอกข้อมูล

ทดลองรัน

จะเห็นว่าการเขียนโปรแกรมสำหรับคอนโทรลจะเป็นไปตามความต้องการของผู้ใช้ (ตามที่เรากำหนด) ขอให้ป้อนโปรแกรมตามที่กล่าวมานี้ต่อเติมในไฟล์ NMECRD2.PAS แล้วทดลองรัน ซึ่งผมได้ทดลองดังนี้

- ป้อนข้อมูลจำนวนหนึ่ง ยกเลิกข้อมูลในลิสต์แล้วโหลดมาใหม่ เพื่อดูว่าข้อมูลเรียงลำดับอักษรหรือไม่

- ยกเลิกข้อมูลในลิสต์ (จะไม่ยกเลิกก็ได้) แล้วป้อนข้อมูลต่อ

- ยกเลิกข้อมูลในลิสต์แล้วโหลดมาใหม่ ทดลองถอนและแก้กับให้แสดงข้อมูลที่แก้ใหม่

หากต้องการดูว่าเมื่อป้อนสตริงที่กรอบป้อนของ comboB แล้ว กรอบคอมโบจะดำเนินการอย่างไร จะต้องป้อนข้อมูลจนรายการในลิสต์ล้นกรอบ คือปรากฏภาพสโครลบาร์แล้วป้อนอักษรแรกของรายการที่ไม่เห็นในกรอบขณะนั้น ถ้าคุณใช้ไมโครซอฟต์วินโดวส์ไทยเอดิชัน สามารถป้อนข้อมูลเป็นภาษาไทยได้

สำหรับตอนต่อไป

เมื่อทดลองรันและเก็บลงไฟล์แล้ว ขอให้ก๊อบปี้ลงไฟล์ NMECRD3.PAS (ใช้ Save As ก็ได้ครับ) เพื่อต่อเติมให้เก็บข้อมูลลงไฟล์ได้ ซึ่งผมจะต้องขอนำไปแทรกในตอนต่อไป อันเป็นเรื่องของไดอะล็อกบ็อกซ์ สำหรับโปรแกรมในตอนนี้รวมทั้ง NMECRD3.PAS ผมได้ฝากมาทางซอฟต์เมล์หากต้องการครับ